MANIFIESTO TECNOLÓGICO · OFICINA DEL DIRECTOR CIENTÍFICO

Computación neuromórfica, lógica memristiva y un tejido de eventos para RXos

De la visión a la ingeniería ejecutable: prototipo software validado y medido en banco.

Rogex Laboratories — Dirección Científica (CSO) y Arquitectura de Sistemas Principal
Ámbito: RXos (Rogex OS) · rogexgreen.com · Rev 0.2

> **Qué cambia frente a Rev 0.1.** La Rev 0.1 fijó la visión. Esta Rev 0.2 la aterriza: incorpora un prototipo *host-side* que fue inspeccionado contra esta rama y **ejecutado y medido**. El manifiesto distingue ahora, con números, lo demostrable en software (asincronía por eventos, plasticidad) de lo que sigue siendo objetivo de hardware.

validado (software) · memristor · spiking neural networks · event fabric · STDP · remote sensing

RESUMEN

La arquitectura von Neumann convierte el cómputo en una máquina de mover datos cuyo coste energético domina sobre el trabajo útil, con externalidad ecológica a escala de gigavatios. Rogex propone la transición hacia sustratos *memristivos* y modelos de *impulsos* que fusionan almacenamiento y cómputo y gastan energía solo ante el evento. Esta Rev 0.2 añade la *validación ejecutable*: un prototipo software que implementa el tejido de eventos y la plasticidad memristiva, medido en banco, y una hoja de ruta concreta para portarlo a RXos.

NOTA DE ESTADO CIENTÍFICO — HONESTIDAD DE INGENIERÍA

VALIDADO (software): demostrado y medido en banco (§5). **OBJETIVO DE I+D (hardware):** memristores físicos, crossbars analógicos, cómputo en memoria — aún no existen en Rogex. **x86 sigue siendo x86:** el sustrato permanece con reloj; «consumo cero» significa ausencia de sondeo en espacio de usuario, no un cero termodinámico. El RXos actual (v4) es un SO bare-metal von Neumann.

---

1 Executive Summary & Manifesto

1.1 · Misión y régimen de rendimientos decrecientes

Rogex existe para construir cómputo que no cueste un planeta. El fin del escalado de Dennard desacopló la densidad de transistores del presupuesto de potencia: empaquetamos más lógica pero no podemos alimentarla ni refrigerarla al mismo ritmo (*dark silicon*), justo cuando la demanda de cómputo crece de forma superlineal. La huida hacia adelante —más datacenters, más gigavatios— es ecológicamente insostenible.

1.2 · El coste ecológico de von Neumann

Separar procesamiento y memoria por un bus finito produce el **cuello de botella de von Neumann**: el sistema gasta su energía **moviendo bits, no computándolos**.

> *La operación útil es ruido energético frente al coste de acarrear los pesos de un lado a otro del bus.*

Horowitz (ISSCC 2014, 45 nm): una suma FP32 cuesta `~0,9 pJ`; leer esos bits de DRAM, `~640 pJ` — factor **~700×**. Escalado: consumo a escala de gigavatios; refresco perpetuo de DRAM (gasta por no olvidar); y una cola ambiental poco contabilizada — **estrés hídrico** y **contaminación acústica** (enfriadoras en decenas de dB sobre el ruido de fondo, 24/7).

1.3 · Tesis central

> *La computación sostenible no se logra apagando servidores ecocidas más rápido, sino construyendo sistemas que, como la biología, gasten energía solo cuando y donde ocurre algo.*

Tres principios: **cómputo en memoria**, **asincronía por eventos** y **localidad de ultra-bajo consumo**. El §5 demuestra que el segundo ya es medible hoy, en software.

---

2 Paradigma neuromórfico y lógica memristiva

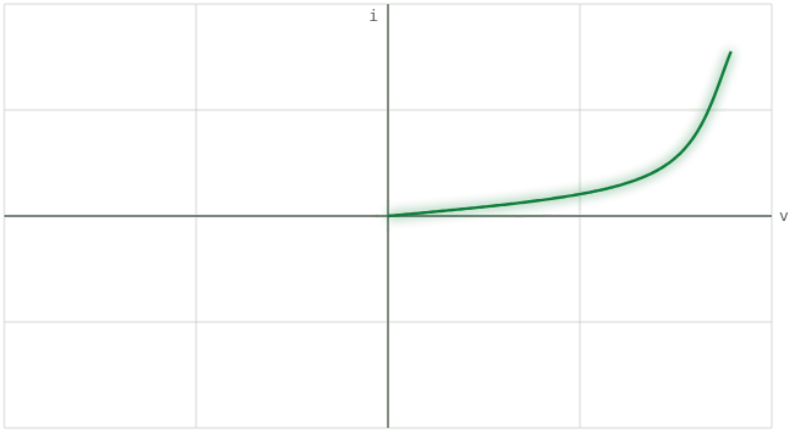
2.1 · El memristor: resistencia con memoria

Cuarto elemento pasivo (Chua, 1971; HP Labs, 2008). La **memristancia** depende de la integral histórica de la corriente, en una variable de estado interna `w`:

$$v(t) = M(w(t)) \cdot i(t) \qquad dw/dt = f(w, i)$$

En una celda de TiO_{2x}, la **distribución de vacantes de oxígeno** gobierna la conductancia; el campo provoca su **migración iónica**. La **no volatilidad** —las vacantes quedan ancladas sin alimentación— hace que el estado **persista años con cero vatios de retención**, eliminando el refresco de DRAM.

FIG. 1 — FIRMA I–V DEL MEMRISTOR



Histéresis pellizcada. El lazo I–V **pasa siempre por el origen**: la resistencia depende del *estado*, no del instante. Es la prueba experimental de la memristancia.

2.2 · La matriz sináptica: el crossbar

En rejilla, los memristores materializan una **matriz de pesos** donde la multiplicación matriz-vector ocurre **en el dominio físico, O(1)** — Ohm pondera, Kirchhoff suma:

$$I_i = \Sigma_j G_{ij} \cdot V_j \qquad \text{— por física, no por instrucciones}$$

2.3 · SNN, asincronía y STDP

Las **SNN** usan neuronas LIF que **solo emiten un pico —y consumen— al cruzar el umbral**. En reposo permanecen latentes. La **STDP** aprende localmente según el orden temporal de los picos. *Anclaje:* TrueNorth (10^6 neuronas, decenas de mW), Loihi (órdenes de magnitud de eficiencia), SpiNNaker (millones de núcleos). El sustrato existe.

---

3 Arquitectura objetivo para RXos

> **EL ERROR A EVITAR**
>
> No es «un SO de sondeo que sustituir por una SNN». RXos **ya** es event-driven: los IRQ de teclado/ratón encolan mensajes IPC y la CPU hace `hlt` en reposo. El trabajo es **generalizar esa base**, no reemplazarla.

3.1 · De la base actual a un tejido de eventos tipado

Diagnóstico de esta rama: base event-driven sólida, pero scheduler en stub, un único flujo, y un camino pseudo-periódico —Rogex Wired— servido por un solo *idle hook* (`wired_service()` drenando `net_poll()` tras despertares del PIT). El refactor correcto, en orden:

1. **Generalizar el IPC en un tejido de eventos tipado** (colas MPSC acotadas, prioridad, destino, contrapresión).
2. **Convertir recepciones del NIC en eventos:** el ISR de virtio encola `RX_EVENT_NET_RX_READY`; un actor de red drena el anillo. Y **eliminar la espera activa de TX** en `vnet_send()` — publicar, marcar `WAITING_TX`, reactivar por compleción.
3. **Sustituir el único idle hook** por fuentes/actores registrados.
4. **Activación por umbral solo para servicios blandos y diferibles.**
5. **Preservar deterministas** los caminos de IRQ, memoria, cripto y *deadlines*.

3.2 · La frontera crítica (no negociable)

> **LÍMITE DE SEGURIDAD**
>
> Un umbral neuronal **jamás** debe decidir si RXos atiende un fallo de página, reconoce una interrupción, completa una escritura a disco, atiende un *watchdog* o aplica un control de acceso. La activación tipo SNN pertenece a batching adaptativo, refresco de UI, agregación de anomalías, telemetría ambiental o pistas de enrutado. La corrección del kernel no se somete a votación estadística.

3.3 · Puerto a C tras validar en Python (fases)

Fase A — tejido de eventos, sin SNN aún. El ISR solo reconoce el dispositivo y encola un descriptor (nunca parsea tramas en contexto de interrupción):

```c
/* kernel/event/event.h */
typedef struct {
    uint32_t type;
    uint16_t source;
    uint16_t destination;
    uint8_t  priority;
    uint8_t  flags;
    uint16_t payload_len;
    uint64_t timestamp_ticks;
    int32_t  stimulus_q16;     /* punto fijo Q16.16 */
    uint8_t  payload[48];
} rx_event_t;
```

- **Fase B — red:** idle hook → evento `RX_EVENT_NET_RX_READY`; mantener un *deadline* de baja frecuencia hasta verificar MSI-X/INTx en todas las máquinas QEMU.
- **Fase C — actores por umbral:** potencial en punto fijo, umbral, refractario y callback acotado; **decaimiento perezoso** al llegar el evento — nunca un «tick neuronal» periódico.

- **Fase D — persistencia/telemetría:** pesos a RXFS solo en *checkpoints*; exportar snapshots a un proceso host. **No** embeber FastAPI/Python en la imagen bare-metal.

La **plasticidad lógica** (redundancia descentralizada + reconfiguración de rutas por ensayo y error vía STDP) es el objetivo de tolerancia a fallos: el sistema *rodea* el defecto en lugar de estrellarse. Plena solo sobre el crossbar; aproximable con el tejido de actores.

---

4 Matriz de impacto — Rogex Green

4.1 · Proyección de paradigma + medición real

> **PROYECCIÓN CONCEPTUAL, NO MEDICIÓN DE RXOS**
>
> La tabla ilustra el diferencial de paradigma anclándose en silicio neuromórfico real. Las cifras *medidas* del §5 validan el **mecanismo** del ahorro (no gastar CPU en reposo), no la energía de un hardware que aún no existe.

DIMENSIÓN	VON NEUMANN	NEUROMÓRFICO (OBJETIVO)
Localización del cómputo	CPU y DRAM separadas por bus	**Cómputo en memoria (crossbar)**
Coste dominante	Movimiento de datos (~700×)	**Conmutación de picos**
Memoria en reposo	Refresco perpetuo de DRAM	**No volátil · `0 W` retención**
Escala de potencia (ref.)	10^{2}–10^{3} W	**10^{-2}–10^{-1} W (Loihi/TrueNorth)**
Diferencial de eficiencia	1× (base)	**10^{2}–10^{4}× en cargas dispersas***
Modo de fallo	Colapso (pantalla azul)	**Degradación graciosa + reconfiguración**

4.2 · Software descentralizado como resistencia climática

- **IA local democratizada:** inferencia en el dispositivo, sin permiso, conexión ni el datacenter de un tercero.
- **Sensórica ambiental (*remote sensing*):** nodos que duermen meses y despiertan solo ante el evento crítico —un umbral de **contaminación acústica en dB**, una firma de deforestación—, procesan *in situ* y transmiten solo la conclusión.
- **Integración honesta en el panel:** el prototipo conserva el contrato de sondeo de 30 s de Vercel (no un WebSocket por visitante), degrada con `Promise.allSettled`, usa token solo-servidor y **etiqueta el bloque como `SOFTWARE SIMULATION`**. Regla: no mezclar energía sináptica simulada con energía medida de datacenters ni datos ambientales reales.

5 Validación ejecutable

El prototipo `rxos_neuromorphic_blueprint` es un paquete Python de **cero dependencias en su núcleo** que modela el paradigma en software. Fue inspeccionado contra esta rama y **ejecutado y medido** en banco. Dos piezas: `MemristorSynapse` (sinapsis memristiva software: `R(x)=R_on·x+R_off·(1−x)`, deriva por historia de corriente, STDP por pares con trazas de decaimiento perezoso, persistencia JSON atómica) y `SpikeDispatcher` (actores tipo SNN, una cola acotada por neurona, *workers* suspendidos en `await queue.get()` — sin sondeo—, LIF por umbral, contrapresión).

MEDICIÓN HEADLINE — EL MECANISMO DEL AHORRO

Con 200 actores en reposo durante 2 s: **0,04 % de CPU** (event-driven) frente a **99,6 % de CPU** en una línea base de sondeo equivalente (63,7 millones de iteraciones desperdiciadas). Es CPU de proceso medida, no energía de hardware — la afirmación defendible.

8/8	0,08 %	~65 k	99,6 %
tests OK (memristor + dispatcher, Python 3.11)	CPU · 1000 neuronas en reposo, 3 s	eventos/s de rendimiento (100 k)	CPU de la línea base de sondeo (vs 0,04 %)

5.1 · Juicio técnico del prototipo

Es **ingeniería honesta y competente**, el complemento exacto de este manifiesto: código defensivo, la reformulación correcta del problema (tejido de eventos + actores, no «x86 se vuelve neuromórfico»), la frontera crítica bien trazada, y **se niega a sobre-afirmar** (mide CPU de proceso, no energía, y lo dice). Decisiones de librería sensatas (asyncio en el núcleo; Brian2/Nengo fuera del kernel; descarta Lava —archivado— y BindsNET por licencia). Se adopta como **banco de validación oficial** de la línea neuromórfica de RXos.

OBSERVACIÓN MENOR

El `pyproject.toml` pide `requires-python >=3.12` (por el extra Brian2), pero el núcleo corre sin problemas en 3.11. Recomendación: relajar el requisito del núcleo a `>=3.11` y aislar 3.12 al extra `research`.

---

§ Apéndices y referencias

Tooling ratificado

asyncio (núcleo) · **Brian2/Nengo** (validación, fuera del kernel) · **FastAPI/Prometheus** (telemetría local) · **perf / RAPL** (medición). Descartados como dependencia núcleo: **BindsNET** (AGPL/PyTorch) y **Lava** (archivado). No afirmar ahorro energético solo desde el % de CPU.

Referencias

1. L. O. Chua, *«Memristor — The Missing Circuit Element»*, IEEE TCT, 1971.
2. D. B. Strukov *et al.*, *«The missing memristor found»*, Nature, 2008.
3. M. Horowitz, *«Computing's Energy Problem…»*, ISSCC, 2014.
4. P. Merolla *et al.*, *«A million spiking-neuron integrated circuit»* (TrueNorth), Science, 2014.
5. M. Davies *et al.*, *«Loihi: A Neuromorphic Manycore Processor»*, IEEE Micro, 2018.
6. W. Maass, *«Networks of spiking neurons: the third generation…»*, Neural Networks, 1997.